

## 周报

### SimHash 的实现、结果分析和未来工作

---

#### 一. SimHash 的实现

String1: **the cat sat on the mat**

String2: **the cat sat on a mat**

String3: **we all scream for ice cream**

:simhash 算法，以上述三个文本为例，整个过程可以分为以下几步：

选择 simhash 的位数为 128 位，主要考虑 md5 生成 128 位哈希值

1): 计算 simHash 码

a). 字符串 String 两两分词得到 **tokens**，作为原始文本中的特征。比如对于"the cat sat on the mat"，采用两两分词的方式得到如下结果：{"th", "he", "e ", " c", "ca", "at", "t ", " s", "sa", " o", "on", "n", " t", " m", "ma"}; 再依据 tokens 在 string 中出现的次数作为特征的权重。

b). 计算每个 tokens 的 128 位 Hash 码;

c). 按 tokens 的 Hash 码的位进行标记，是 1 则标记为正的权重值、否则标记为负的权重值;

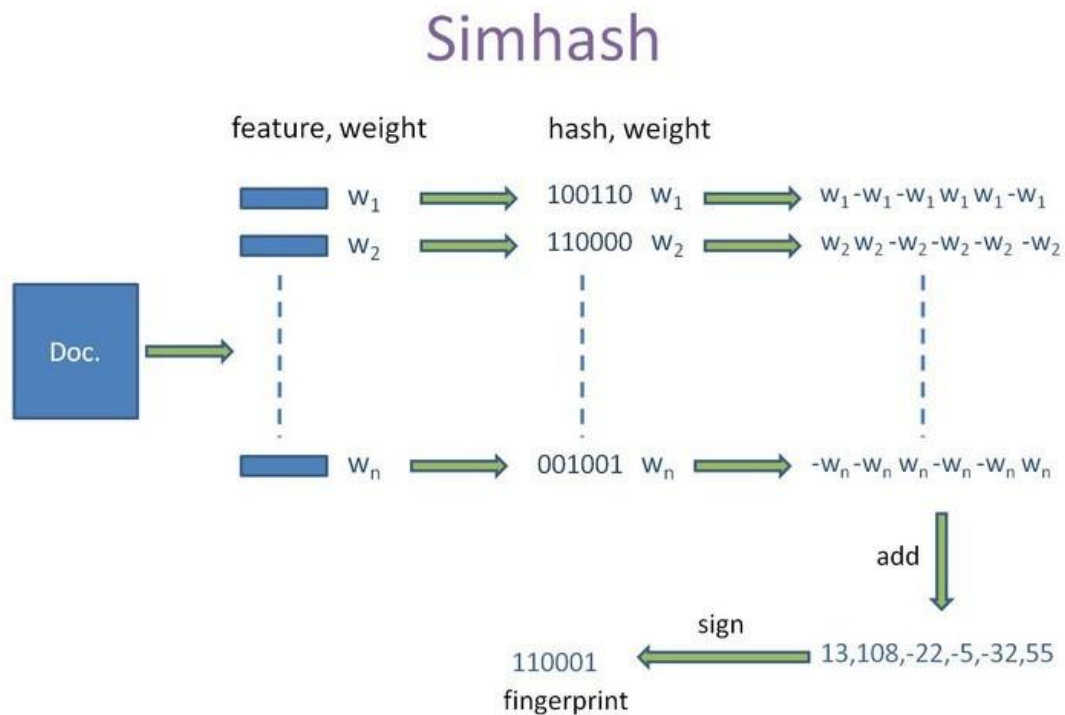
d). 把每个 tokens 的 Hash 码按位进行统计求和;

e). 进行签名, 大于 0 则为 1, 否则为 0, 得到 128 位 simHash

指纹。

2): 计算两个 simHash 码的汉明距离,

给出 simHash 的 int 值(unsigned char): 先做异或, 然后统计异或后二进制位数为 1 的个数



---

## 详解

The processing string is : abef 第一个字符串

两两分词作为特征, 即 feature1 = "ab" feature2 = "ef"

以子串在原始串中出现的次数作为其权重

Current string is : ab 对应的权重是 1

Corresponding md5 hash value in binary string is : 128 位 md5 哈希值的二进制表示如下

000110000111111011110100010000110110000100100010110100011100

11000010111101000000

110111000010101110010010111100001110101110100000

Current string is : ef 对应的权重是 1

Corresponding md5 hash value in binary string is : 128 位 md5 哈希值的

二进制表示如下

11111101011011110001100110000100101100010111101110001110110

10000110011100110101

010011110000000111000010001011011011111001000011

每个子串, 依据自己的 md5 哈希值以及对应的权重, 若二进制值为 1,

则该位加上权重; 为 0, 则减去其权重。得到两个加权表示如下:

Vector\_hash[0][0] to Vector\_hash[0][127] is :

-1-1-111-1-1-1-1111111-11111-11-1-1-11-1-1-1-111-111-1-1-1-11-1-11-1-

1-11-111-11

-1-1-1111-1-111-1-1-1-11-11111-11-1-1-1-1-1-111-1111-1-1-1-11-11-111

1-1-11-1-11-

11111-1-1-1-1111-11-1111-11-1-1-1-1-1

Vector\_hash[1][0] to Vector\_hash[1][127] is :

1111111-11-111-11111-1-1-111-1-111-1-1-1-11-1-11-111-1-1-11-11111-11

11-1-1-1111-

111-11-1-1-1-111-1-1111-1-111-11-11-11-1-11111-1-1-1-1-1-1111-1-1-

1-11-1-1-11-

111-111-111111-1-11-1-1-1-111

Add result is : 对所有特征（子串）的加权表示按位累加

0002200-200220220200002-2-202-2-2-2-220-22000-2-200-220000022-2

0-2002020-220-2-2

-202-20222-2000-20-2002-202200-2-20-20-20220-20-2-22-2002000-202

0202020000-2-2-2

00

Final fingerprint is : 上述累加加过，逐位遍历，若值>1，则设为 1；  
否则设为 0，得到最终的指纹，亦即 simhash 值

00011000011011000010000101000010000000100000010010000011000

10010111001000000000000110010100000000100000100000100010101

0100000000

The processing string is : abcd 第二个字符串

Current string is : ab

Corresponding md5 hash value in binary string is :

000110000111111011110100010000110110000100100010110100011100

11000010111101000000

110111000010101110010010111100001110101110100000

Current string is : cd

Corresponding md5 hash value in binary string is :

011010000110010110101110101100111010100111101101001010001111  
10011010011110011110

110001000101010010110010010110011110010111010000

Vector\_hash[0][0] to Vector\_hash[0][127] is :

-1-1-111-1-1-1-1111111-11111-11-1-1-11-1-1-1-111-111-1-1-1-11-1-11-1-  
1-11-111-11

-1-1-1111-1-111-1-1-1-11-11111-11-1-1-1-1-1-111-1111-1-1-1-11-11-111  
1-1-11-1-11-

11111-1-1-1-1111-11-1111-11-1-1-1-1-1

Vector\_hash[1][0] to Vector\_hash[1][127] is :

-111-11-1-1-1-111-1-11-111-11-1111-11-111-1-1111-11-11-1-11111-111-1  
1-1-11-11-1-

1-111111-1-111-11-1-11111-1-11111-111-1-1-11-1-1-11-11-11-1-11-111-1  
-11-1-11-111

-1-11111-1-11-1111-11-1-1-1-1

Add result is :

-20002-2-2-2-222002002020020-20000-2-222002-20-2-22002-20000000  
00-2-20220020-200

-22-2022200-20000-222-2002-2-2-200000002-202-2-22-202020-2-20222

-200022000-2-2-2

-2

Final fingerprint is :

000100000010011000100101110000001000010000000100000000000000  
100111110010000000000000100011000000000100100100001010100001  
1100000001

计算汉明距离(验证正确)

The Hamming Distance between the current strings is : 26

计算两个字符串相似度

$(128 - 26) / 128$  用的是 128 位哈希

Likeness: 79.6875

---

## 二. SimHash 的结果分析

### 实例 1

The processing string is : cdab

The processing string is : abcd

The Hamming Distance between the current strings is : 0

Likeness: 100

请按任意键继续...

---

## 实例 2

The processing string is : abcd

The processing string is : abcd

The Hamming Distance between the current strings is : 0

Likeness: 100

请按任意键继续...

---

## 实例 3

The processing string is : the cat sat on the mat

The processing string is : the cat sat on a mat

The Hamming Distance between the current strings is : 16

Likeness: 87.5

请按任意键继续...

---

## 实例 4

The processing string is : we all scream for ice cream

The processing string is : the cat sat on a mat

The Hamming Distance between the current strings is : 51

Likeness: 60.1563

请按任意键继续...

---

注意：

对于短文本应用不好，准确率不高。另外，特征的定义，权重的设置，以及分词策略都对结果影响明显.

例如

"Hierarchical object representation Object subdivision Hierarchical clustering of levels of detail Classifies"

"Hierarchical spatia representation Satial subdivision Hierarchical clustering of levels of detail Classifies"

这样两个文本，假如采用逐词分割抽取特征，结果较好。同样，为了在两两分词的策略下取得较好结果，在从文本上看看似大体比较相近的情况下，必须保证两两分词取得的特征也是近似的。所以，上述文本才会出现‘spatia’‘satial’这样的不正确词汇。如果使用正确的词汇，spatial 等，那么两两分词取得的特征不一致，纵使两段文本总体上看比较近似，但是，海明距离相对还是过大。所以，特征抽取和权重定义显得特别重要，应用到体数据时，更要注意

同样地

"the cat sat on a mat"

"the cat sat on the mat"

在程序中，海明距离达到16-20，相似度为84.5%-87.5%



而

"the cat sat on a mat"

"the cat sat on b mat"

相似度为90%，海明距离是12，

"on a matthe cat sat "

"the cat sat on a mat"

两者相似度100% 海明距离0

而

"on a mat the cat sat "

"the cat sat on a mat"

两者相似度53.9%，海明距离59

进一步说明上述推断

---

### 三. 未来工作

未来工作的总体结构和思路如下：

相似哈希位数取 128，相似判别的海明距离 k 取 15

1. 对体数据分块，得到 n 个 blocks
2. 每个 block 里的体数据值作为 string，将之分割，每一个体素值作为特征，其在当前 string 出现的次数作为对应的权重
3. 对每个 block，计算 string 的 md5 哈希值，并计算 simhash，得到

最终的 fingerprint

4. 一个体数据最终得到  $n$  个 fingerprint 构成的集合  $Q$
5. 由于相似哈希位数取 128，相似判别的海明距离  $k$  取 15，所以将待分类的 fingerprint 分成 16 段，每段 8 位
6. 拷贝  $Q$  至 16 份， $Q_0 \sim Q_{15}$

```
for( $Q_i : i \text{ 属于 } [0,15]$ )
```

```
{
```

```
    for( $Q_i$  所有 fingerprint)
```

```
    {
```

```
        置换当前 fingerprint 的第  $i*8$  到  $(i + 1)*8 - 1$  位到整个 fingerprint 的最前面
```

```
    }
```

```
        将当前拷贝  $Q_i$  进行排序，得到有序的 fingerprint 集合,以便快速查找
```

```
}
```

7. 对于每一个待分类的 fingerprint，同样拷贝 16 份，对应于 16 个拷贝，做同样的置换操作，然后并行进行匹配，寻找所有前 8 位与之相同的指纹
8. 对于 7 中所找到的所有 fingerprint，计算其与待检测 fingerprint 是否至多有 15 位不同，若是，归为一类
9. 构建码表，获取解码端所需数据